REMARKS

Claims 1-38 are pending. Claims 1 and 20 are independent.

Applicant acknowledges with thanks the examiner's indication that claims 7, 12, 15, 18, 19, 26, 31, 34, 37 and 38 would be allowable.

The examiner noted that the declaration of inventor Donald F. Hooper is defective because the inventor's country of citizenship was omitted. Submitted herewith (as Exhibit A) is a replacement declaration signed by the inventor Donald F. Hooper in which Mr. Hooper's citizenship is identified.

The examiner objected to the drawings under 37 CFR § 1.83(a) on the ground that the provisions of 37 CFR § 1.83(a) require that the drawings show every feature of the invention, and that therefore the features of claims 1, 7-11 must be shown in the drawings, or the feature(s) cancelled from the claims. Applicant submits that Applicant's figures comply with 37 CFR § 1.83(a) for reasons discussed below.

35 USC § 113 states "[t]he applicant shall furnish a drawing where necessary for the understanding of the subject matter to be patented." Further, as provided by 37 CFR § 1.81(a), "[t]he applicant for a patent is required to furnish a drawing of his or her invention where necessary for the understanding of the subject matter sought to be patented."

Thus, the main statutory provisions of the Patent Act regarding drawings, as well as rule 1.81(a), make it clear that it is not necessary to show in the drawings every feature of the claims as specified in the claims. Rather, it is only when the subject matter in the claims cannot be understood without the assistance of drawings that drawings become necessary.

Additionally, in discussing the requirements of 37 CFR § 1.83(a), MPEP 608.02(d) notes "[a]ny structural detail that is of sufficient importance to be described should be shown in the drawings" (emphasis added). Thus, the Patent Office's own interpretation of the provisions of 37 CFR § 1.83(a) makes it clear that furnishing drawings corresponding to the features in the claims is not a mandatory requirement (as indicated by the use of the discretionary language "should be shown" in MPEP 608.02(d)).

Nevertheless, the drawings comply with 37 CFR § 1.83(a). Independent claim 1 is directed to a method for operating a processor. FIG. 1 shows a processor. The method includes receiving data, and then loading data into a group of bits of a register associated with the processing thread according to the processing thread number. Thus, the details of the elements of claim 1 are sufficiently clear that they can be understood without the use of drawings. Applicant also notes that page 10, lines 25-26, of the application shows an embodiment of an instruction that causes the processor to perform the method. The applicant, therefore, submits that it is not necessary, in this case, to have a drawing illustrating these additional features of independent claim 1.

Claims 7-11 describe additional features of the method described in claim 1, including, for example, specifying how "loading" is performed (see claim 7), and the format of the token of the FAST_WR instruction. Applicant notes that the table at page 11, lines 20, to page 12, line 10, of the application provides details showing the options and parameters associated with the method. Thus, the features of claim 7-11 are readily understood without the use of drawings.

The examiner rejected claims 1-2, 5-6, 8-10, 20-21, 24-25 and 27-29 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,446,190 to Barry. The examiner also rejected claims 1-2, 6, 8-9, 20-21, 25 and 27-28 under 35 USC § 102(e) as being anticipated by U.S. Patent No. 6,279,066 to Velingker. Further, the examiner rejected claims 3-4, 11, 13-14, 16-17, 22-23, 30, 32-33 and 35-36 under 35 USC § 103(a) as being unpatentable over Barry.

Specifically, with respect to the examiner's rejection of independent claim 1 on the basis of Barry, the examiner stated:

> 8. Referring to claim 1, Barry has taught a method of operating a processor comprising;
> a) receiving data specified by execution of a fast-write instruction in a processing thread identified by a processing thread number. See column 9, lines 52-58, and Fig.7A. Note that the fast-write instruction is equivalent to the LIM (Load Immediate) instruction. Also, Barry mentions that context switches occur. Therefore, the system includes multiple threads, and threads are inherently numbered in some way so that they may be identified when switching occurs.
> b) the fast-write instruction further specifying a register, the register having multiple groups of bits, each group of bits associated with one of multiple thread available on the processor. See Fig.7A, and column 9, lines 32-36, and note that the LIM instruction specifies a register to be

> loaded. The register has multiple groups of bits, the first group being the upper 16 bits, and the second group being the lower 16 bits. Since a thread will include the LIM instruction (threads comprise instructions), each group of bits will be associated with the current thread, which is one of multiple threads.
> c) selecting a group of bits associated with the processing thread, the group of bits being selected from the multiple groups of bits of the register specified by the fast-write instruction according to the processing thread number. See Fig.7A, and note that if thread 0 (0 being the thread number) includes a LIM instruction that has LOC bits equal to 00 or 01, then either the upper or lower group of bits will be selected. This selection is done according to processing thread number as thread 0 contains the LIM instruction. That is, in response to thread 0 being executed, a group of bits from the register will be selected (due to the thread having the LIM instruction).
> d) loading the data into the selected bit positions of the register. See Fig.7A, and note that when LOC = 00 or 01, the group is loaded with the immediate data supplied in the LIM instruction. (emphasis added, Office Action, pages 4-5, paragraph 9)

Applicant's independent claim 1 recites "receiving data specified by execution of a fast-write instruction in a processing thread identified by a processing thread number, the fast-write instruction further specifying a register, the register having multiple groups of bits, each group of bits associated with one of multiple threads available on the processor; selecting a group of bits associated with the processing thread, the group of bits being selected from the multiple groups of bits of the register specified by the fast-write instruction according to the processing thread number." Thus, which particular group of bits, from the multiple groups of bits of a register (e.g., a control and status register), is selected to have data loaded thereto is performed according to the particular processing thread number. In other words, the identity of the processing thread is determinative to the selection of the group of bits of the register being operated upon.

In contrast, Barry describes register file indexing (RFI) techniques that provide indirect control of register addressing in a Very Long Word (VLIW) processor (col. 1, lines 18-20). Barry explains that RFI control specification is performed through RFI control registers. Barry further explains that a 'load immediate' instruction is used to load data to the registers participating in the RFI operation. With respect to the LIM instruction, Barry explains:

> **First, control information as illustrated in FIG. 6 for each register file port is written into an RFI control register 810 and 910 by use of a load immediate (LIM) instruction 700 whose encoding format is shown in FIG. 7A and whose syntax/operation 710 is shown in FIG. 7B. The LIM**

instruction 700 is first used to load MRFXAR halfword H1410 of FIG. 4 to set up the desired extension RFI control register to be mapped to MRFXDR2303 in FIG. 3A. Then, the LIM instruction loads a data value to the desired control register by using the address for MRFXDR2. Each halfword section of a control register is loaded separately by definition of the LIM instruction.

For purposes of clarity, the LIM data path from instruction register 804 H0 halfword bits 15-0 is not shown. This data path is selectively controlled to load the H0 halfword of the LIM instruction to either the low or high halfword portion of any of the MRF extension registers listed in FIG. 5. For example, a LIM instruction could cause the loading of its H0 halfword to the H1 portion of the RFIAM0 register 520 of FIG. 5. In reference to the common arithmetic RFI port control logic of FIG. 8, one of the three control portions of RFIAM0 would be located in an update control register 0 for that port, such as 810, for, in this case, the ALU 852. In a similar manner, the other two port control values would be loaded into their own port update control register 0s contained in their own RFI port control logic. Other ManArray instructions can load the RFI control registers through use of the MRF data bus 809. The MRF data bus 809 is also used for saving the RFI port registers, for example, during a context switch operation. The specific LIM instruction description is as follows. The halfword form of the LIM instruction loads a 16-bit immediate value into the upper halfword (H1) or lower halfword (H0) of an SP or PE target register Rt. The 16-bit immediate value is interpreted as a sign "neutral" value, meaning that any value in the range -32768 to 65535 is accepted. This covers the 2's complement signed value range of -32768 to +32767 and the unsigned value range of 0 to 65535. (FIG. 7A, and col. 9, lines 25-62)

Thus, the immediate value specified in the LIM instruction is loaded into the RFI control register specified in the LIM instruction Rt5-0 field. The immediate value is loaded to either the low or high word portion of the specified RFI control register based on the values indicated in the LOC field of the LIM instruction.

Particularly, and as shown in FIG. 7A, when the two-bit LOC field is set to '00', the immediate value is loaded into the high portion (H1) of the destination register, with the low portion (H0) remaining unaffected. When LOC is set to '01', the immediate value is loaded into H0, with H1 remaining unaffected. When LOC is set to '10', the immediate value is loaded into H0, with H1 being set to '0x0000', and when LOC is set to '11' the immediate value is loaded to H0 with H1 being set to '0xFFFF'. Thus, which group of bits of the destination register is loaded with the LIM instruction's immediate value depends solely on the specified value of the LOC field of the LIM instruction which presumably is set by a programmer. The LOC value is not representative of a thread number, nor does it have anything to do with the particular identity

of any threads. Thus, Barry does not teach that the selection of the particular group of bits (i.e., H0 or H1) of the destination register depends on the thread number. In fact, the identity of any of the contexts of the processor (be it a running context or a dormant context) is entirely irrelevant to selecting the group of bits of the destination register into which the immediate value is to be loaded.

Therefore, contrary to the examiner's contentions, Barry fails to disclose or suggests at least the feature of "selecting a group of bits associated with the processing thread, the group of bits being selected from the multiple groups of bits of the register specified by the fast-write instruction according to the processing thread number," as required by applicant's independent claim 1. Applicant's independent claim 1 is therefore patentable over Barry.

As noted, the examiner also rejected independent claim 1 as being anticipated by Velingker. Specifically, the examiner stated:

> 17. Referring to claim 1, Velingker has taught a method of operating a processor comprising:
> a) receiving data specified by execution of a fast-write instruction in a processing thread identified by a processing thread number. See Figs.2-4 and column 5, lines 19-50. Note from Fig.3 that multiple processors each execute different threads (threads 1, 2, and 3). When a thread wants access to a shared resource, it issues an instruction (fast-write instruction) to set a bit in the RNC (register)
> b) the fast-write instruction further specifying a register, the register having multiple groups of bits, each group of bits associated with one of multiple thread available on the processor. See Figs.2-3, column 5, lines 19-50, and column 6, lines 10-18. In one embodiment of the invention, the resource negotiation cell (RNC) is a register with three groups of bits, each group comprising two bits (a request bit and a completion bit). Each group is associated with a different thread. When a thread wants to request a resource or indicate access completion, a fast-write instruction including data is issued to set an appropriate bit in the corresponding group.
> c) selecting a group of bits associated with the processing thread, the group of bits being selected from the multiple groups of bits of the register specified by the fast-write instruction according to the processing thread number. Again, see column 5, lines 19-50, column 6, lines 10-18, and Figs.2-3. When a thread wants to access a shared resource or indicate access completion, the thread's group of bits is selected for modification.
> d) loading the data into the selected bit positions of the register. See column 5, lines 19-50, column 6, lines 10-18, and Figs.2-3. When the thread wants to access a shared resource, it will write a '1' into a bit of the group of bits. And, when it is finished accessing the resource, it will

> write a T into another bit of the group of bits. (Office Action, pages 6-7, paragraph 17)

Applicant respectfully disagrees with the examiner's contentions.

Velingker describes a resource negotiation mechanism to arbitrate between a plurality of accessing processing agents for access to a common resource such as a register or a memory location (Abstract, col. 1, lines 15-18). Velingker's apparatus includes a shared resource negotiator (SRN) 100 and three processors (102, 104 and 106) seeking to access a common resource such as shared control register (FIG. 1, and col. 3, lines 49-52). Velingker explains that:

> **In particular, FIG. 1 shows a shared resource negotiator 100 including three access request bits 160, 162, 164 for writing by each of the three processors 102-106, and a corresponding three grant bits 170, 172, 174 for reading by the same three processors 102-106. The processors 102, 104 and 106 include appropriate software which allows the respective processor to access the shared control register 150 only after first requesting access to the shared control register 150 by writing to the corresponding request bit 160, 162 or 164 and reading a successful grant value in the corresponding grant bit 170, 172 or 174. (Col. 3, lines 55-65)**

Velingker further explains:

> **FIG. 3 shows that according to the present embodiment, each of the processors 102-106 can write only to its assigned bit location within the shared resource negotiator 100 and not to the other bit locations in the shared resource negotiator 100. For example, the first processor 102 can write only to bit 0 and not to bits 1 and 2 of the shared resource negotiator 100, the second processor 104 can write only to bit 1 and not to bits 0 and 2, and the third processor 106 can write only to bit 2 and not to bits 0 and 1. However, in the disclosed embodiment, all three of the processors 102-106 can read all bits of the shared resource negotiator 100, providing an additional source of information to the accessing processors 102-106 about the granting of access to the corresponding shared resource to other processors as well as to the corresponding processor.**
>
> **FIG. 4 shows a process of requesting permission to write to a shared resource, e.g., the shared control register 150 shown in FIG. 1, in accordance with the principles of the present invention.**
>
> **In particular, FIG. 4 shows in step 402 that a processor desiring to write to a shared resource first lodges a request for the shared resource by writing a predetermined logic level, e.g., a `1', to its assigned request bit in the corresponding shared resource negotiator 100.**

> **In step 404, the requesting processor reads back the grant status (i.e., access granted or access denied) from the assigned read bit in the shared resource negotiator 100. In the disclosed embodiment the write and read bit are in the same relative location in the shared resource negotiator 100 register.**
>
> **Step 406 determines whether or not the grant bit read back in step 404 indicates access granted (e.g., `1') or access denied (e.g., `0'). If the requesting processor reads back a logic value indicating that access has been granted, e.g., a `1', the requesting processor will then presume that it has been granted permission to write to the shared resource and complete its access to the shared resource in step 408. In the disclosed embodiment, the processor granted permission to access the shared resource will retain that permission until the corresponding request bit is cleared, e.g., by a write from the corresponding processor. (Col. 5, lines 26-67)**

Thus, a processor that accesses a common resource sends a request for the resource by setting a corresponding request bit on the SRN 100. In other words, which of the SRN's bits are set depends on which processor wishes to access the shared resource. At no point does Velingker describe that any of the SRN bits are set in accordance with "a thread" or "a thread number." Indeed, Velingker does not even mention threads. Applicant notes that a "thread", as understood by the person of average skill in the art, means "[i]n programming, a part of a program that can execute independently of other parts. Operating systems that support multithreading enable programmers to design programs whose threaded parts can execute concurrently" (Webopedia.com). Applicant contends that the meaning of the term "thread" certainly does not mean "processor". Therefore, Velingker fails to disclose or suggest at least the feature of "selecting a group of bits associated with the processing thread, the group of bits being selected from the multiple groups of bits of the register specified by the fast-write instruction according to the processing thread number," as required by applicant's independent claim 1.

Furthermore, as noted, the various processors connected to Velingker's apparatus are configured to set corresponding bits in the SRN 100. Those bits are accessible independently by each of the corresponding processors, and are not arranged in a register. Thus, because Velingker does not describe a register, Velingker does not disclose or suggest "the fast-write instruction further specifying a register, the register having multiple groups of bits, each group of bits associated with one of multiple threads available on the processor," nor does it disclose or

suggest "loading the data into the bit positions of the selected group of bits of the register," as required by applicant's independent claim 1.

Moreover, Velingker explicitly states that "[i]n particular, FIG. 4 shows in step 402 that a processor desiring to write to a shared resource first lodges a request for the shared resource by writing a predetermined logic level, e.g., a `1`, to its assigned request bit in the corresponding shared resource negotiator 100" (col. 5, lines 46-50). Velingker indicates that the bits of the SRN 100 are set by hardwiring the processors to those bits, and having a processor assert a logical 1 when vying for a shared resource. Velingker neither describes nor suggests that the various bits of the SRN 100 are written or loaded by execution of any type of an instruction (e.g., processor based instruction), let alone the execution of a fast write instruction. Accordingly, Velingker also fails to disclose or suggest at least "receiving data specified by execution of a fast-write instruction in a processing thread identified by a processing thread number, the fast-write instruction further specifying a register," as required by applicant's independent claim 1.

In view of the foregoing, applicant's independent claim 1 is patentable over the art cited by the examiner.

Rejected claims 2-6, 8-11, 13-14 and 16-17 depend from independent claim 1, and are therefore patentable over the cited art for at least the same reasons as independent claim 1.

Independent claim 20 recites the features of "receive data specified in the fast-write instruction in a processing thread identified by a processing thread number, the fast-write instruction further specifying a register, the register having multiple groups of bits, each group of bits associated with one of multiple threads available on the computer; select a group of bits associated with the processing thread, the group of bits being selected from the multiple groups of bits of the register specified in the fast-write instruction according to the processing thread number; and load the data into the bit positions of the selected group of bits of the register." For reasons similar to those provided with respect to independent claim 1, at least these features are not disclosed by the cited art. Accordingly, independent claim 20 is patentable over the cited art. Rejected claims 21-25, 27-30, 32-33 and 35-36 depend from independent claim 20 and are therefore patentable for at least the same reasons as independent claim 20.

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

In view of the foregoing remarks, applicant respectfully submits that the application is in condition for allowance and such action is respectfully requested at the examiner's earliest convenience.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

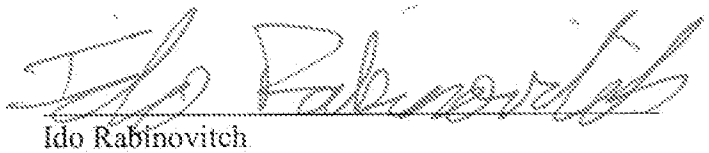
Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

Please apply any required fees to deposit account 06-1050, referencing the attorney docket number shown above.

Respectfully submitted,

Date: Feb 22, 2007



Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

PTO Customer No. 20985
Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906